## AMENDMENTS TO THE CLAIMS

1. (Currently amended) A method of ~~performing native binding to execute native code during the translation of~~translating subject program code executable by a subject processor into target program code executable by a target processor, ~~wherein native code is code executable by the target processor,~~ said method comprising:

dividing the subject program code into a plurality of subject program code units;

translating one or more of the subject program code units into one or more target program code units; and

executing the one or more target program code units on the target processor;

wherein the translating step includes identifying ~~certain~~ a subject function in the subject program code having a corresponding native function of native code[[;]], wherein the native code is code executable by the target processor, and identifying the native function of the native code which corresponds to the identified subject function~~subject program code~~; and

wherein the executing step includes executing the ~~corresponding native code~~ native function on the target processor instead of executing a translated version of the identified ~~subject program code~~ subject function, including transforming zero or more function parameters from a target code representation to a native code representation, invoking the native function with the transformed zero or more function parameters according to a prototype of the native function, and transforming zero or more return values of the invoked native function from a native code representation to a target code representation.

2. (Cancelled)

3. (Cancelled)

4. (Currently amended) The method of claim ~~3~~1, wherein at least one of the transformations in the transforming steps generates an intermediate representation of the transformation.

5. (Currently amended) The method of claim ~~3~~1, wherein at least one of the transformations in the transforming steps generates target code.

6. (Currently amended) The method of claim ~~3~~1, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

7. (Currently amended) The method of claim ~~3~~1, wherein the native function executing step comprises:

transforming a function parameter from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and

transforming a result of the invoked native function from a native code representation to a target code representation.

8. (Currently amended) The method of claim ~~3~~1, wherein the function parameter transforming step and the native function invoking step are described in subject code by translator specific instructions added to the subject instruction set.

9. (Currently amended) The method of claim 1, wherein the steps of identifying the ~~certain subject code~~subject function and its corresponding native ~~code~~ function are performed using a bind point description.

10. (Cancelled)

11. (Currently amended) The method of claim ~~10~~9, further comprising inserting in the target code a call stub to the native function during translation of the subject code when encountering the subject function contained in the bind point description.

12. (Original) The method of claim 9, wherein the bind point description is embedded within a translator performing the translation.

13. (Original) The method of claim 9, further comprising reading the bind point description from a stored bind point description file at the beginning of translation execution.

14. (Currently amended) The method of claim 9, wherein the bind point description includes a location in the subject code and a corresponding native function, wherein the location in the subject code identifies the ~~certain subject program code~~ subject function having a corresponding native ~~code~~ function and the native function identifies the corresponding native ~~code~~function.

15. (Currently amended) The method of claim 9, wherein the bind point description includes a location in the subject code and a reference to code to be invoked, wherein the location in the subject code identifies the ~~certain subject program code~~ subject function having a corresponding native ~~code~~ function and the reference to code to be invoked identifies the corresponding native ~~code~~function.

16. (Original) The method of claim 15, wherein the code to be invoked is target code.

17. (Original) The method of claim 9, wherein the bind point description includes a native function call which is inserted in the target code either before, after, or in place of a subject function call.

18. (Original) The method of claim 9, further performing runtime symbol patching comprising:

encoding subject-to-native function mappings in a symbol table of the subject program,

replacing entries in the symbol table of the subject program with special native binding markers, and

interpreting the special native binding markers when encountered during translation as bind point descriptions to identify an appropriate native function to call.

19. (Currently amended) The method of claim 9, wherein the bind point description includes a correspondence to an external ~~Schizo~~ call command, wherein the ~~Schizo~~ external call command is a translator-specific native binding instruction, the method comprising:

when encountering a bind point description identifying an external ~~Schizo~~ call command during translation of the subject code, diverting the flow of translation to the execution of the external ~~Schizo~~ call command.

20. (Currently amended) The method of claim 19, wherein the external ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and

generating an intermediate representation of the external ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

21. (Currently amended) The method of claim 19, wherein the external ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and

generating target code for the external ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

22. (Currently amended) The method of claim 1, further comprising:

inserting ~~Schizo~~ external call commands into the subject code, wherein external ~~Schizo~~ call commands are translator-specific native binding instructions; and

detecting the ~~Schizo~~ external call commands during translation of the subject code.

23. (Currently amended) The method of claim 22, further comprising:

when encountering ~~a Schizo~~ the external call command during translation of the subject code, diverting the flow of translation to the execution of the ~~Schizo~~ external call command.

24. (Currently amended) The method of claim 23, wherein the external~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and

generating an intermediate representation of the ~~Schizo~~ external call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

25. (Currently amended) The method of claim 23, wherein the ~~Schizo~~ external call command execution step comprises:

interpreting the ~~Schizo~~ external call command; and

generating target code for the ~~Schizo~~ external call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function

26. (Currently amended) The method of claim 22, wherein the ~~Schizo~~ external call commands are variable length instructions including multiple sub-component instructions.

27. (Currently amended) The method of claim 26, wherein the multiple sub-component instructions include an ~~Schizo~~ Escape sub-component instruction, said ~~Schizo~~ external call commands detecting step further comprising detecting the ~~Schizo~~ Escape sub-component instruction.

28. (Currently amended) The method of claim 27, wherein said ~~Schizo~~ Escape sub-component instruction further identifies a type of ~~Schizo~~ external call command represented by the other sub-component instructions of the ~~Schizo~~ external call command.

29. (Original) The method of claim 1, further comprising:

parsing and decoding a native binding implementation scripting language containing native binding scripts;

interpreting the native binding scripts during translation;

generating an intermediate representation of the native binding scripts to transform a function parameter from a target code representation to a native code representation.

30. (Original) The method of claim 29, further comprising:

integrating the intermediate representation of the native binding scripts into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

31. (Original) The method of claim 1, further comprising:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

interpreting the native code call stub function; and

generating an intermediate representation of the native code call stub function binding scripts to transform a function parameter from a target code representation to a native code representation.

32. (Original) The method of claim 21, further comprising:

integrating the intermediate representation of the native code call stub function into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

33. (Currently amended) The method of claim ~~3~~1, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers;

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

34. (Original) The method of claim 1, further comprising:

parsing a scripting language implementation of a native code call stub function;

compiling the parsed native code call stub function into a native code executable module; and

linking the native code executable module with an executable for performing the translation.

35. (Original) The method of claim 34, wherein the native code executable module is executable for:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

36. (Currently amended) The method of claim 34, wherein the steps of identifying the ~~certain subject code~~subject function and its corresponding native ~~code~~ function are performzed using a bind point description, said bind point description including a subject function and a native code call stub function, wherein the subject function identifies the ~~certain subject program code~~ subject function having corresponding native ~~code~~ function and the native code call stub function identifies the corresponding native ~~code~~function.

37. (Original) The method of claim 36, further comprising encoding the identity of the native function of the native code call stub function in the scripting language implementation of the native code executable module.

38. (Currently amended) The method of claim ~~3~~1, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a target code call stub function with the transformed subject registers;

invoking from the target code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

39. (Original) The method of claim 38, further comprising:

generating an intermediate representation of the native function executing step;

integrating the intermediate representation of the native function executing step into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

40. (Original) The method of claim 1, wherein the subject function to be executed is a system call.

41. (Original) The method of claim 1, wherein the subject function to be executed is a library function.

42. (Currently amended) A computer-readable storage medium having software resident thereon in the form of computer-readable code executable by a computer having a target processor to perform the steps of:

dividing a subject program in subject program code executable by a subject processor into a plurality of subject program code units;

translating one or more of the subject program code units into one or more target program code units of target program code; and

executing the one or more target program code units on the target processor;

wherein the translating step includes identifying ~~certain~~ a subject function in the subject program code having a corresponding native function of native code[[;]], wherein the native code is code executable by the target processor, and identifying the native function of the native code which corresponds to the identified subject function~~subject program code~~; and

wherein the executing step includes executing the ~~corresponding native code~~ native function on the target processor instead of executing a translated version of the identified ~~subject program code~~ subject function, including transforming zero or more function parameters from a target code representation to a native code representation, invoking the native function with the transformed zero or more function parameters according to a prototype of the native function, and transforming zero or more return values of the invoked native function from a native code representation to a target code representation.

43. (Cancelled

44. (Cancelled)

45. (Currently amended) The computer-readable storage medium of claim [[44]]42, wherein at least one of the transformations in the transforming steps generates an intermediate representation of the transformation.

46. (Currently amended) The computer-readable storage medium of claim [[44]]42, wherein at least one of the transformations in the transforming steps generates target code.

47. (Currently amended) The computer-readable storage medium of claim [[44]]42, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

48. (Currently amended) The computer-readable storage medium of claim [[44]]42, wherein the native function executing step comprises:

transforming a function parameter from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and

transforming a result of the invoked native function from a native code representation to a target code representation.

49. (Currently amended) The computer-readable storage medium of claim [[44]]42, wherein the function parameter transforming step and the native function invoking step are described in subject code by translator specific instructions added to the subject instruction set.

50. (Currently amended) The computer-readable storage medium of claim 42, wherein the steps of identifying the ~~certain~~ subject ~~code~~ function and its corresponding native ~~code~~ function are performed using a bind point description.

51. (Cancelled)

52. (Currently amended) The computer-readable storage medium of claim [[51]]50, wherein said computer readable code is further executable for inserting in the target code a call stub to the native function during translation of the subject code when encountering the subject function contained in the bind point description.

53. (Original) The computer-readable storage medium of claim 50, wherein the bind point description is embedded within a translator performing the translation.

54. (Original) The computer-readable storage medium of claim 50, wherein said computer readable code is further executable for reading the bind point description from a stored bind point description file at the beginning of translation execution.

55. (Currently amended) The computer-readable storage medium of claim 50, wherein the bind point description includes a location in the subject code and a corresponding native function, wherein the location in the subject code identifies the ~~certain subject program code~~ subject function having a corresponding native ~~code~~ function and the native function identifies the corresponding native ~~code~~function.

56. (Currently amended) The computer-readable storage medium of claim 50, wherein the bind point description includes a location in the subject code and a reference to code to be invoked, wherein the location in the subject code identifies the ~~certain subject program code~~subject function having a corresponding native ~~code~~ function and the reference to code to be invoked identifies the corresponding native ~~code~~function.

57. (Original) The computer-readable storage medium of claim 56, wherein the code to be invoked is target code.

58. (Original) The computer-readable storage medium of claim 50, wherein the bind point description includes a native function call which is inserted in the target code either before, after, or in place of a subject function call.

59. (Original) The computer-readable storage medium of claim 50, wherein said computer readable code is further executable for performing runtime symbol patching comprising:

encoding subject-to-native function mappings in a symbol table of the subject program,

replacing entries in the symbol table of the subject program with special native binding markers, and

interpreting the special native binding markers when encountered during translation as bind point descriptions to identify an appropriate native function to call.

60. (Currently amended) The computer-readable storage medium of claim 50, wherein the bind point description includes a correspondence to an external ~~Schizo~~ call command, wherein the ~~Schizo~~ call command is a translator-specific native binding instruction, for:

when encountering a bind point description identifying an external ~~Schizo~~ call command during translation of the subject code, diverting the flow of translation to the execution of the external ~~Schizo~~ call command.

61. (Currently amended) The computer-readable storage medium of claim 60, wherein the external ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and

generating an intermediate representation of the external ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

62. (Currently amended) The computer-readable storage medium of claim 60, wherein the external ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and

generating target code for the external ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

63. (Currently amended) The computer-readable storage medium of claim 42, wherein said computer readable code is further executable for performing the following steps:

inserting Schizo call commands into the subject code, wherein ~~Schizo~~ call commands are translator-specific native binding instructions; and

detecting the ~~Schizo~~ call commands during translation of the subject code.

64. (Currently amended) The computer-readable storage medium of claim 63, wherein said computer readable code is further executable for performing the following steps:

when encountering a ~~Schizo~~ call command during translation of the subject code, diverting the flow of translation to the execution of the ~~Schizo~~ call command.

65. (Currently amended) The computer-readable storage medium of claim 64, wherein the ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and

generating an intermediate representation of the ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

66. (Currently amended) The computer-readable storage medium of claim 64, wherein the ~~Schizo~~ call command execution step comprises:

interpreting the ~~Schizo~~ call command; and

generating target code for the ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function

67. (Currently amended) The computer-readable storage medium of claim 63, wherein the ~~Schizo~~ call commands are variable length instructions including multiple sub-component instructions.

68. (Currently amended) The computer-readable storage medium of claim 67, wherein the multiple sub-component instructions include a ~~Schizo~~ Escape sub-component instruction, said ~~Schizo~~ call commands detecting step further comprising detecting the ~~Schizo~~ Escape sub-component instruction.

69. (Currently amended) The computer-readable storage medium of claim 68, wherein said ~~Schizo~~ Escape sub-component instruction further identifies a type of ~~Schizo~~ call command represented by the other sub-component instructions of the ~~Schizo~~ call command.

70. (Original) The computer-readable storage medium of claim 42, wherein said computer readable code is further executable for performing the following steps:

parsing and decoding a native binding implementation scripting language containing native binding scripts;

interpreting the native binding scripts during translation;

generating an intermediate representation of the native binding scripts to transform a function parameter from a target code representation to a native code representation.

71. (Original) The computer-readable storage medium of claim 70, wherein said computer readable code is further executable for performing the following steps:

integrating the intermediate representation of the native binding scripts into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

72. (Original) The computer-readable storage medium of claim 42, wherein said computer readable code is further executable for performing the following steps:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

interpreting the native code call stub function; and

generating an intermediate representation of the native code call stub function binding scripts to transform a function parameter from a target code representation to a native code representation.

73. (Original) The computer-readable storage medium of claim 62, wherein said computer readable code is further executable for performing the following steps:

integrating the intermediate representation of the native code call stub function into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest

74. (Currently amended) The computer-readable storage medium of claim [[44]]42, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers;

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

75. (Original) The computer-readable storage medium of claim 42, wherein said computer readable code is further executable for performing the following steps:

parsing a scripting language implementation of a native code call stub function;

compiling the parsed native code call stub function into a native code executable module; and

linking the native code executable module with an executable for performing the translation.

76. (Original) The computer-readable storage medium of claim 75, wherein the native code executable module is executable for:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

77. (Currently amended) The computer-readable storage medium of claim 75, wherein the steps of identifying the certain ~~subject code~~subject function and its corresponding native ~~code~~ function are performed using a bind point description, said bind point description including a subject function and a native code call stub function,

wherein the subject function identifies the ~~certain subject program code~~ subject function having a corresponding native ~~code~~ function and the native code call stub function identifies the corresponding native ~~code~~function.

78. (Original) The computer-readable storage medium of claim 77, wherein said computer readable code is further executable for encoding the identity of the native function of the native code call stub function in the scripting language implementation of the native code executable module.

79. (Original) The computer-readable storage medium of claim 44, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a target code call stub function with the transformed subject registers;

invoking from the target code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

80. (Original) The computer-readable storage medium of claim 79, wherein said computer readable code is further executable for performing the following steps:

generating an intermediate representation of the native function executing step;

integrating the intermediate representation of the native function executing step into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

81. (Original) The computer-readable storage medium of claim 42, wherein the subject function to be executed is a system call.

82. (Original) The computer-readable storage medium of claim 42, wherein the subject function to be executed is a library function.

83. (Currently amended) A computer apparatus, comprising:

a target processor; and

a memory containing translator code executed by the target processor to perform the steps of:

dividing a subject program in subject program code executable by a subject processor into a plurality of subject program code units;

translating one or more of the subject program code units into one or more target program code units of target program code; and

executing the one or more target program code units on the target processor;

wherein the translating step includes identifying ~~certain~~ a subject function in the subject program code having a corresponding native function of native code[[;]], wherein the native code is code executable by the target processor, and identifying the native function of the native code which corresponds to the identified subject function~~subject program code~~; and

wherein the executing step includes executing the ~~corresponding native code~~native function on the target processor instead of executing a translated version of the identified ~~subject program code~~subject function, including transforming zero or more function parameters from a target code representation to a native code representation, invoking the native function with the transformed zero or more function parameters according to a prototype of the native function, and transforming zero or more return values of the invoked native function from a native code representation to a target code representation.

84. (Cancelled)

85. (Cancelled)

86. (Currently amended) The apparatus of claim [[85]]83, wherein at least one of the transformations in the transforming steps generates an intermediate representation of the transformation.

87. (Currently amended) The apparatus of claim [[85]]83, wherein at least one of the transformations in the transforming steps generates target code.

88. (Currently amended) The apparatus of claim [[85]]83, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

89. (Currently amended) The apparatus of claim [[85]]83, wherein the native function executing step comprises:

transforming a function parameter from a target code representation to a native code representation;

invoking the native function with the transformed function parameter according to a prototype of the native function; and

transforming a result of the invoked native function from a native code representation to a target code representation.

90. (Currently amended) The apparatus of claim [[85]]83, wherein the function parameter transforming step and the native function invoking step are described in subject code by translator specific instructions added to the subject instruction set.

91. (Currently amended) The apparatus of claim 83, wherein the steps of identifying the ~~certain subject code~~subject function and its corresponding native ~~code~~ function are performed using a bind point description.

92. (Cancelled)

93. (Previously presented)The apparatus of claim 92, said translator code further comprising code executable by said target processor for inserting in the target code a call stub to the native function during translation of the subject code when encountering the subject function contained in the bind point description.

94. (Currently amended) The apparatus of claim 91, wherein the bind point description is embedded within ~~a~~ the translator code performing the translation.

95. (Previously presented) The apparatus of claim 91, said translator code further comprising code executable by said target processor for reading the bind point description from a stored bind point description file at the beginning of translation execution.

96. (Currently amended) The apparatus of claim 91, wherein the bind point description includes a location in the subject code and a corresponding native function, wherein the location in the subject code identifies the ~~certain subject program code~~subject function having a corresponding native ~~code~~ function and the native function identifies the corresponding native ~~code~~function.

97. (Currently amended) The apparatus of claim 91, wherein the bind point description includes a location in the subject code and a reference to code to be invoked, wherein the location in the subject code identifies the ~~certain subject program code~~subject function having a corresponding native ~~code~~ function and the reference to code to be invoked identifies the corresponding native ~~code~~function.

98. (Previously presented) The apparatus of claim 97, wherein the code to be invoked is target code.

99. (Previously presented) The apparatus of claim 91, wherein the bind point description includes a native function call which is inserted in the target code either before, after, or in place of a subject function call.

100. (Previously presented) The apparatus of claim 91, said translator code further comprising code executable by said target processor for performing runtime symbol patching comprising:

encoding subject-to-native function mappings in a symbol table of the subject program,

replacing entries in the symbol table of the subject program with special native binding markers, and

interpreting the special native binding markers when encountered during translation as bind point descriptions to identify an appropriate native function to call.

101. (Currently amended) The apparatus of claim 91, wherein the bind point description includes a correspondence to an external ~~Schizo~~ call command, wherein the ~~Schizo~~ call command is a translator-specific native binding instruction, the method comprising:

when encountering a bind point description identifying an external ~~Schizo~~ call command during translation of the subject code, diverting the flow of translation to the execution of the external ~~Schizo~~ call command.

102. (Currently amended) The apparatus of claim 101, wherein the external ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and generating an intermediate representation of the external ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

103. (Currently amended) The apparatus of claim 101, wherein the external ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and

generating target code for the external ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

104. (Currently amended) The apparatus of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

inserting ~~Schizo~~ call commands into the subject code, wherein ~~Schizo~~ call commands are translator-specific native binding instructions; and

detecting the ~~Schizo~~ call commands during translation of the subject code.

105. (Currently amended) The apparatus of claim 104, said translator code further comprising code executable by said target processor for performing the following steps:

when encountering a ~~Schizo~~ call command during translation of the subject code, diverting the flow of translation to the execution of the ~~Schizo~~ call command.

106. (Currently amended) The apparatus of claim 105, wherein the ~~Schizo~~ call command execution step comprises:

interpreting the external ~~Schizo~~ call command; and generating an intermediate representation of the ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function.

107. (Currently amended) The apparatus of claim 105, wherein the ~~Schizo~~ call command execution step comprises:

interpreting the ~~Schizo~~ call command; and

generating target code for the ~~Schizo~~ call command which:

transforms a function parameter from a target code representation to a native code representation, and

invokes the native function with the transformed function parameter according to a prototype of the native function

108. (Currently amended) The apparatus of claim 104, wherein the ~~Schizo~~ call commands are variable length instructions including multiple sub-component instructions.

109. (Currently amended) The apparatus of claim 108, wherein the multiple sub-component instructions include a ~~Schizo~~ Escape sub-component instruction, said ~~Schizo~~ call commands detecting step further comprising detecting the ~~Schizo~~ Escape sub-component instruction.

110. (Currently amended) The apparatus of claim 109, wherein said ~~Schizo~~ Escape sub-component instruction further identifies a type of ~~Schizo~~ call command represented by the other sub-component instructions of the ~~Schizo~~ call command.

111. (Previously presented) The apparatus of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

parsing and decoding a native binding implementation scripting language containing native binding scripts;

interpreting the native binding scripts during translation;

generating an intermediate representation of the native binding scripts to transform a function parameter from a target code representation to a native code representation.

112. (Previously presented) The apparatus of claim 111, said translator code further comprising code executable by said target processor for performing the following steps:

integrating the intermediate representation of the native binding scripts into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

113. (Previously presented) The apparatus of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers according to a uniform call stub interface;

interpreting the native code call stub function; and

generating an intermediate representation of the native code call stub function binding scripts to transform a function parameter from a target code representation to a native code representation.

114. (Previously presented) The apparatus of claim 103, said translator code further comprising code executable by said target processor for performing the following steps:

integrating the intermediate representation of the native code call stub function into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest

115. (Currently amended) The apparatus of claim [[85]]83, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers;

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

116. (Previously presented) The apparatus of claim 83, said translator code further comprising code executable by said target processor for performing the following steps:

parsing a scripting language implementation of a native code call stub function;

compiling the parsed native code call stub function into a native code executable module; and

linking the native code executable module with an executable for performing the translation.

117. (Previously presented) The apparatus of claim 116, wherein the native code executable module is executable for:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a native code call stub function with the transformed subject registers; and

invoking from the native code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

118. (Currently amended) The apparatus of claim 116, wherein the steps of identifying the ~~certain subject code~~subject function and its corresponding native ~~code~~ function are

performed using a bind point description, said bind point description including a subject function and a native code call stub function, wherein the subject function identifies the ~~certain subject program code~~subject function having a corresponding native ~~code~~ function and the native code call stub function identifies the corresponding native ~~code~~function.

119. (Previously presented) The apparatus of claim 118, said translator code further comprising code executable by said target processor for encoding the identity of the native function of the native code call stub function in the scripting language implementation of the native code executable module.

120. (Currently amended) The apparatus of claim [[85]]83, wherein the native function executing step further comprises:

transforming in target code all subject register values from the target code representation to the native code representation;

invoking from target code a target code call stub function with the transformed subject registers;

invoking from the target code call stub function the native function with particular subject registers and/or parameter stack according to the prototype of the native function.

121. (Previously presented) The apparatus of claim 120, said translator code further comprising code executable by said target processor for performing the following steps:

generating an intermediate representation of the native function executing step;

integrating the intermediate representation of the native function executing step into an intermediate representation forest for a block of subject code; and

generating target code for the intermediate representation forest.

122. (Previously presented) The apparatus of claim 83, wherein the subject function to be executed is a system call.

123. (Previously presented) The apparatus of claim 83, wherein the subject function to be executed is a library function.